AD-A055 167 ARINC RESEARCH CORP ANNAPOLIS MD F/G 17/2
APPLICATION OF TACTICAL DATA SYSTEM COMPUTER-PROGRAMMING SPECIF--ETC(U)
SEP 67 C MCINDOE, C KIMME, D MILESON N00024-67-C-1413
UNCLASSIFIED 555-01-4-812 NL

| OF |
AD
A055167

END
DATE
FILMED
7-78
DDC

FOR FURTHER TRAN

Publication 555-01-4-812

(5)

APPLICATION OF TACTICAL DATA SYSTEM COMPUTER-PROGRAMMING SPECIFICATION TO TAOC

September 1967

Prepared for
Naval Electronic Systems Command
Department of the Navy
under Contract N00024-67-C-1413

ARINC RESEARCH CORPORATION

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 555-01-4-812
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): APPLICATION OF TACTICAL DATA SYSTEM COMPUTER-PROGRAMMING SPECIFICATION TO TAOC

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER: 555-01-4-812

(10) 7. AUTHOR(s): C. McIndoe, C. Kimme, D. Mileson, T. Worley

8. CONTRACT OR GRANT NUMBER(s): (15) N0024-67-C-1413

9. PERFORMING ORGANIZATION NAME AND ADDRESS
ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Electronic Systems Command
Department of the Navy

(11) 12. REPORT DATE: Sept 1967

13. NUMBER OF PAGES: 23 P.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Naval Electronic Systems Command
Department of the Navy

(12) 15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

UNCLASSIFIED/UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer-program documentation procedure to replace conventional flow diagrams for tactical data systems is summarized. Two types of graphic documentation are described: Program Flow Diagrams and Functional Flow Charts.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

444247

hC

APPLICATION OF TACTICAL DATA SYSTEM COMPUTER-PROGRAMMING SPECIFICATION TO TAOC

September 1967

Prepared for
Naval Electronic Systems Command
Department of the Navy
under Contract N00024-67-C-1413

by

C. McIndoe
C. Kimme
D. Mileson
T. Worley

ARINC RESEARCH CORPORATION
a subsidiary of Aeronautical Radio, Inc.
2551 Riva Road
Annapolis, Maryland 21401

Publication 555-01-4-812

Prepared under Contract N00024-67-C-1413
which grants to the United States
Government a license to use any
material in this publication for
Government purposes.

ABSTRACT

A computer-program documentation procedure to replace conventional flow diagrams for tactical data systems is summarized. Two types of graphic documentation are described: Program Flow Diagrams and Functional Flow Charts.

CONTENTS

1. INTRODUCTION

Study of the documentation of computer programming for tactical data systems reveals many shortcomings in the flow charts and program listings provided for maintenance and operational use. As a result of these shortcomings, operational and maintenance personnel must often spend long periods of time studying these charts in order to clearly understand system functions -- a great waste of manpower.

To assist in resolving the problem, this report summarizes an approach to computer-program documentation that was first described in ARINC Research's publication Tactical Data System Computer Programming Specification*. The method replaces conventional flow diagrams with the Program Flow Diagrams and Functional Flow Charts discussed in the following chapter. An example of computer-program documentation according to this method is presented in Chapter 3.

*ARINC Research Corporation, Publication 414-04-4-692.

2. DETAILS OF THE DOCUMENTATION METHOD

2.1 Description of the Documents

The Program Flow Diagrams and Functional Flow Charts required by ARINC Research's computer-programming specification use geometric shapes, symbols, and supplementary notations to illustrate, on a single sheet, the logical flow of data and the sequence of operations in a digital computer program, routine, or subroutine. The two types of graphic documentation are described as follows:

Program Flow Diagram - The Program Flow Diagram is a basic document that illustrates the decision processes and the resultant actions in terms of design logic. The design logic includes tests and actions; examples of tests are "Target Closer Than 200 Miles", "Target Displaying IFF", and "Target Confirmed"; examples of actions are "Set Drop Track Bit", "Subtract Range from Previous Range", and "Compute New Target Position". When multiple actions are to be performed after a test or series of tests, optional sequences must be identified to provide flexibility in the programming process. Thus, the programmer can attempt various combinations to improve the efficiency of the program.

Functional Flow Chart - The Functional Flow Chart illustrates the programming process required to satisfy the logic-design requirements of the Program Flow Diagram, and the options selected. In addition to containing blocks for each logical decision and action, the Functional Flow Chart contains blocks for programming operations such as masking, shifting, incrementing, clearing, storing, exclusive ORing, etc. The terminology in the decision and action boxes is an encoding of the respective requirement illustrated on the Program Flow Diagram. For example, where the Program Flow Diagram displays "Set Drop-Track Bit", the Functional Flow Chart displays "Set DT=1", where DT has been defined as the word section or table that contains drop-track information.

2.2 Preparation of the Documents

Program Flow Diagrams and Functional Flow Charts are prepared to illustrate the following program divisions:

- The overall program, containing major routines
- The major routines, containing major subroutines
- The major subroutines

2.2.1 Degree of Detail

The degree of detail is influenced by the problem, by the programming language to be used, and by the level of program subdivision being flow-charted. Flow diagrams that deal with low-level program subdivisions require more detail than flow charts that deal with the higher levels. Lines connecting the boxes illustrate the flow within the diagrams, with separate lines used for each basic flow. When direction of flow is not readily apparent, arrowheads show the direction.

2.2.2 Overall Program

The Overall Program is documented with a Program Flow Diagram containing one block for each major routine. One block is included for each peripheral device which provides an input to, or receives an output from, the computer being programmed. Major routines within the overall program have a single input but may have one, or two, outputs. Major routines are subgrouped to identify subroutines within the major routine, and to illustrate the relationship and flow between the subroutines. A functional flow chart is not normally prepared for the Overall Program.

2.2.3 Major Routines

Major Routines are documented with Program Flow Diagrams containing one block for each subroutine (which is, in turn, illustrated by a separate diagram). These diagrams also contain decision symbols, where applicable, to illustrate decision criteria for entry into the subroutine. Flow lines illustrate flow between the subroutines. Functional Flow Charts may be provided at this program level, depending upon the complexity of the program.

2.2.4 Major Subroutines

Major Subroutines are documented with both Program Flow Diagrams and Functional Flow Charts illustrating, by appropriate programming symbols, the individual decision, the individual action to be performed in the process illustrated by the diagrams, and the tie-ins to the subfunctional diagram(s) that produce the input, and receive the output.

2.2.5 Identification

Major routines are identified by name only. However, each grouping block at each level has an identifying number with a decimal indenture used to indicate subordination of groupings. Numbers between the first and second decimal indicate major subroutines within major routines. Numbers between the second and third decimal indicate subroutines within major subroutines. Additional subordinate numbering is used as required to identify the subordinate groupings within the subroutines.

TABLE 1
EXPLANATIONS OF SYMBOLS

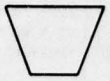
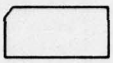
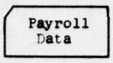
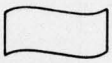
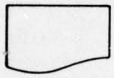
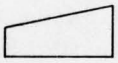
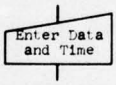
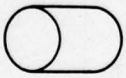
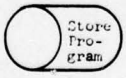
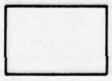
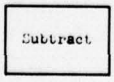
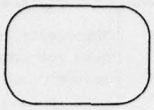
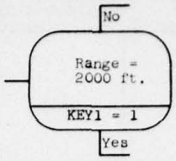
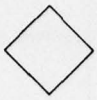
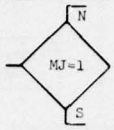
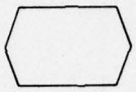
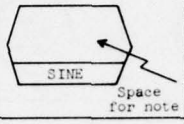
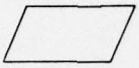
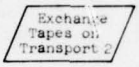
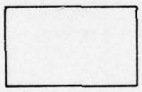
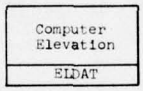
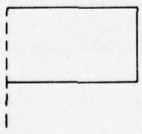
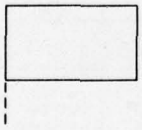
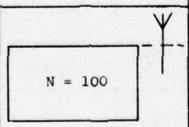
Name	Symbol	Example	Notes
Input/Output Symbols			
Basic			Represents the making available of data for processing (input), or the recording or displaying of processed data (output)
Data Card			Punched cards, card files, etc.
Magnetic Tape			
Punched Tape			
Printed Document			
Manual Input			Data entered manually by on-line keyboards, switch settings, etc. at time of processing
Display			Data displayed by on-line printers, video devices, etc. at time of processing
Mass Storage			Magnetic drums, magnetic discs, etc. with on-line access
Processing Symbols			
Basic			Represents the performance of an operation that changes the value, form, or location of data

TABLE 1 (continued)

Name	Symbol	Example	Notes
Processing Symbols (continued)			
Decision			Informative data, such as the data unit on which the decision is made or a sense-switch setting, may be included in a subdivision of the symbol
Switch and Branch			Data is routed to each of two output paths by setting a switch, either "set" (S) or "normal" (N).
Subroutine			Informative data, such as the subroutine label, may be included in a subdivision of the symbol
Manual Operation			An off-line process geared to the speed of a human
Auxiliary-Equipment Operation			Off-line operation performed on equipment not under direct control of the central processing unit
Nonauxiliary Equipment Operation			On-line operation performed on equipment under direct control of the central processing unit. Informative data, such as names of data being processed, may be included in a subdivision of the symbol.
Annotation Symbols			
Basic			Used for the inclusion of comments or explanatory notes. The broken line is extended, in the most convenient direction, to the appropriate point on the flow line
Assertion			Indicates a special condition exists at a certain point
Insertion			For use in updating; should not appear on final charts
Communication-Link Symbols			
Basic			Represents data transmission from one location to another. Arrows indicate direction of flow

2.2.6 Program Flow Diagram and Functional Flow Chart Symbols

Symbols are used on Program Flow Diagrams and Functional Flow Charts to represent the functions of a data processing program or system. Basic symbols are established for the functions that are ordinarily included in high-level flow charts; i.e., input/output, processing, and annotation. Specialized symbols -- within these categories -- are established for the detailed functions that are ordinarily included in low-level (subroutine) flow charts. The symbols, which are listed and explained in Table 1, are in accordance with MIL-STD-682, Flow Chart Symbols for ADP Systems.

2.3 Flow Lines

Straight lines (vertical or horizontal) are used to show the flow of control or of data between the symbols on a flow diagram. Symbols requiring two output lines normally have both output lines perpendicular to the input line. None of the lines -- input or output -- are considered to be an inherent part of any symbol.

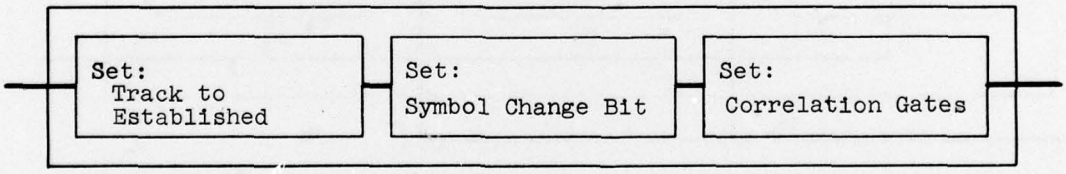
2.3.1 Direction of Flow

Horizontal flow diagramming is required. Directional flow of inputs and outputs is left to right, unless otherwise indicated. A connection of lines to the grouping boxes indicates program sequence. Lines passing left to right through a grouping box indicate that the decision or action listed in the box must be performed before the program can proceed. Lines entering a grouping box from the top, or from the left, with no line leaving the right of the box, indicate an action that must be performed at this time. However, subsequent program operations within this subroutine are not dependent on the results of this action.

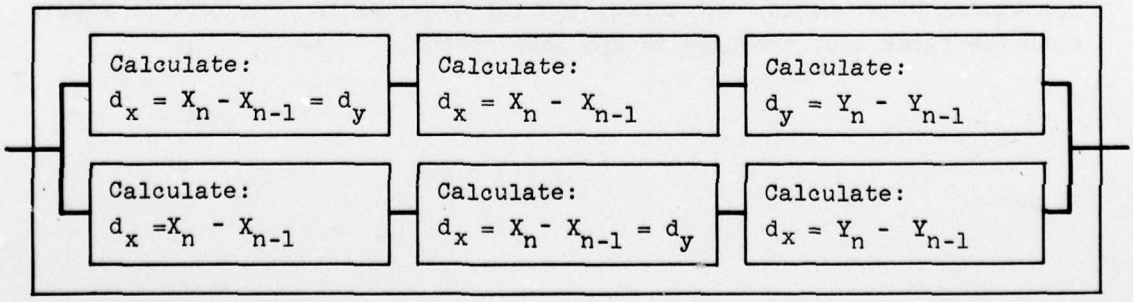
2.3.2 Fixed and Optional Sequences

Fixed sequences and optional sequences are illustrated on the Program Flow Diagram as follows:

(1) Actions that must be performed sequentially:

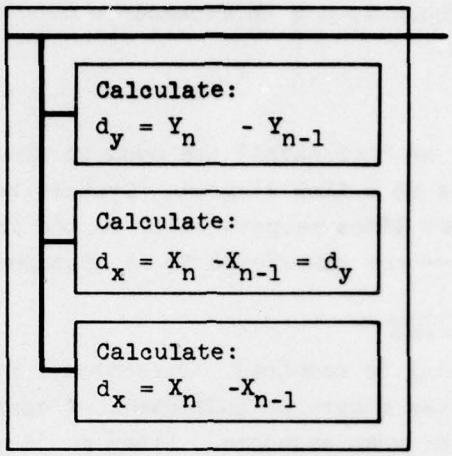


(2) Sequences of actions that may be performed in optional sequence:



Optional sequences are prohibited on Functional Flow Charts since they illustrate the order in which the options have been exercised.

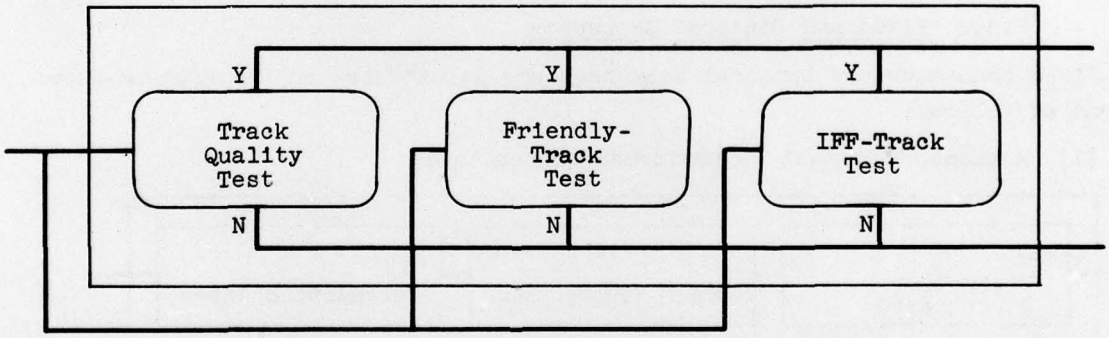
(3) Actions that must be performed at this time in the program but can be performed in any sequence (the person doing the coding may find an opportunity to improve program efficiency by changing the sequence of action):



2.3.3 AND-OR Decisions

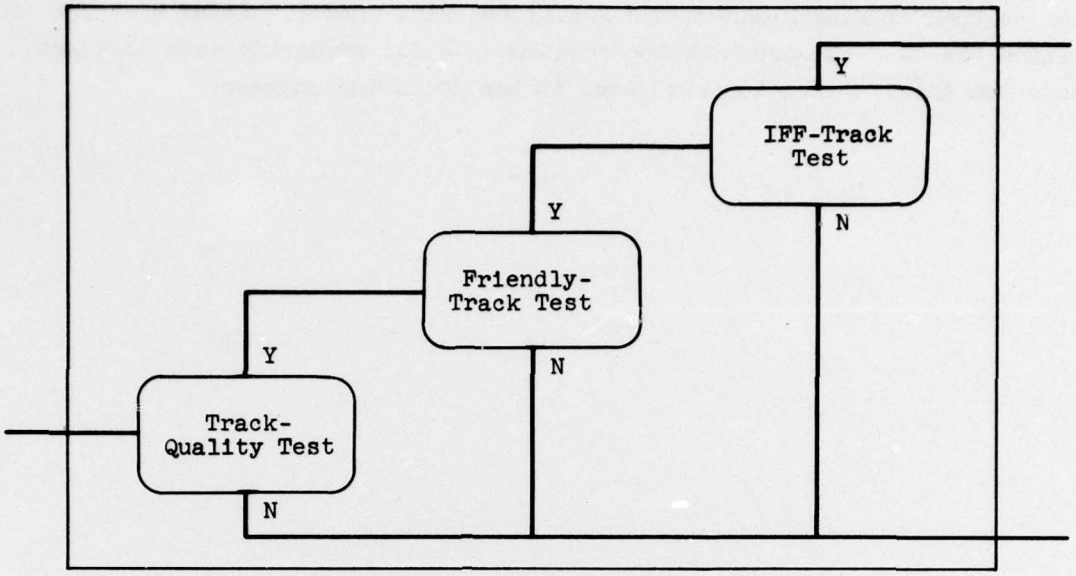
Flow lines illustrate actions associated with AND and OR decision as follows:

(1) OR decision:



On Program Flow Diagrams, the illustration implies optional sequence; on Functional Flow Charts, the actual testing sequence is from left to right, with the first test provided in the left symbol.

(2) AND decision:



2.3.4 Notations

Internal consistency is maintained in Program Flow Diagrams and Functional Flow Charts by adherence to the following rules:

- All text words have initial capitals, with the remaining letters in lower case.
- References to programs and data units, when they are the names used in the computer program, are all in capital letters.
- References to hardware labels appear as found on the hardware.
- No question marks are placed at the end of the text in a decision box; the symbol itself indicates a question.
- Text is condensed to fit within the symbols; abbreviations are avoided where possible.
- Mathematical notation is minimized unless expressing complete equations. Where possible, text is in ordinary English in terms that can be easily understood.
- Questions are phrased (in decision boxes) so that they can be answered with a "Yes" or "No". Other responses are permissible; e.g., >, <, = , and combinations thereof. If the decisions are expressed in English words, the first letter of each word is capitalized.
- Action illustrated by an action box is indicated by a statement of the action in the upper left corner of the box; i.e., Set:, Subtract:, Compute:, Increment:, etc. The action to be performed is indicated on the following lines within the box.

2.3.5 Annotated Listings

Annotated listings describing program implementation are also required in addition to Program Flow Diagrams and Functional Flow Charts. ARINC Research Publication 414-04-4-692 presents the requirements for preparing such listings. Examples from this publication are given in the following chapter.

3. EXAMPLE OF COMPUTER PROGRAM DOCUMENTATION

The following documents are examples of computer documentation introduced in ARINC Research Publication 414-04-4-692:

- Program Flow Diagram (Major Routine)
- Major Subroutine Program Flow Diagram (with Functional Description of Logic Flow)
- Major Subroutine Functional Flow Chart

They represent the type of computer program documentation that replaces the conventional flow chart shown in the Appendix. The three documents are discussed in turn in the following sections.

3.1 Program Flow Diagram (Major Routine)

The Program Flow Diagram shown in Figure 1 provides a visibility of the Major Routine (Tracking) that is not available from the documentation shown in the Appendix. The interrelationships of the 18 major subroutines are also clearly illustrated; the Tracking Routine is identified by name only, whereas each of the major subroutines is identified by name and number.

3.2 Program Flow Diagram (Major Subroutine)

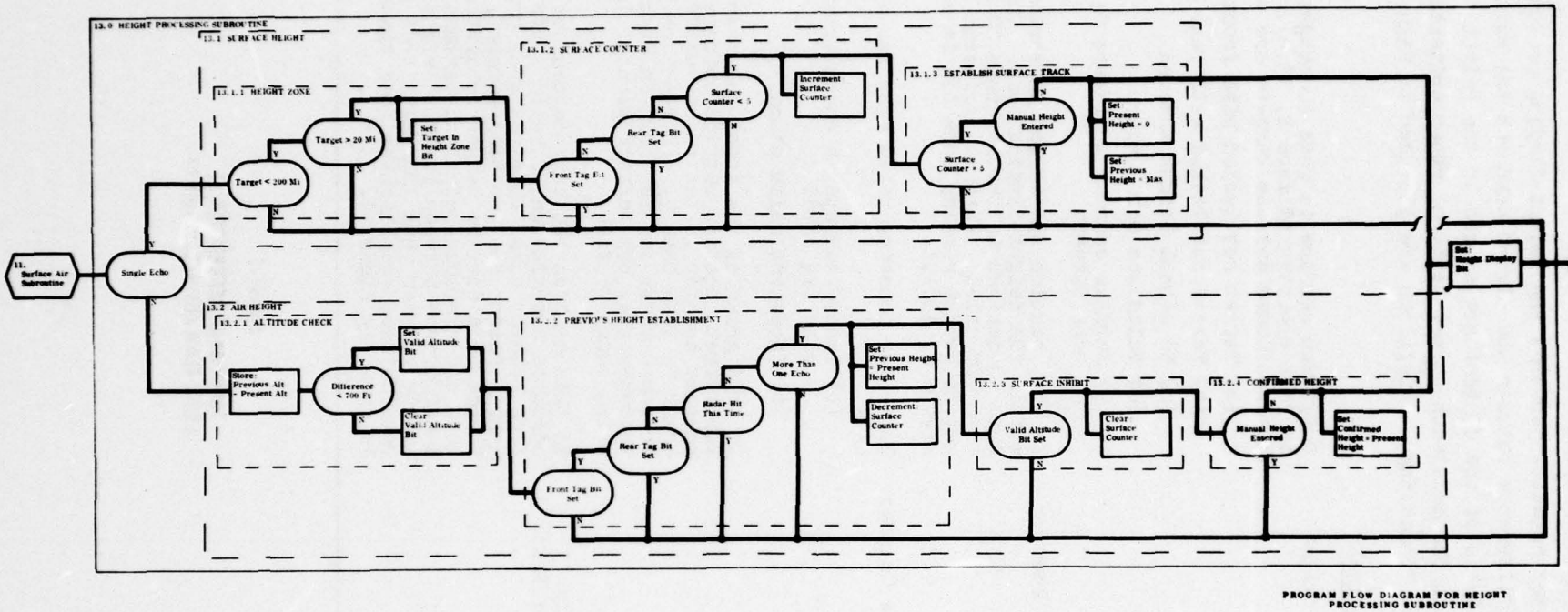
The Program Flow Diagram shown in Figure 2 is an example of a Major Subroutine Flow Diagram; a diagram such as this is prepared for each major subroutine. In this case, the diagram is for subroutine number 13 (Height Processing Subroutine). Subroutine 13.1 describes height processing for surface targets, while 13.2 shows height processing for air targets. Within subroutine 13.1 are sub-subroutines 13.1.1, Height Zone; 13.1.2, Surface Counter; and 13.1.3, Establish Surface Track. Similar sub-subroutines are shown for subroutine 13.2.

Understanding of the program is enhanced by providing brief functional descriptions of each numerically identified subroutine. The functional description of logic flow for subroutine 13 (shown in Figure 3) demonstrates how easily this understanding can be achieved by using such documentation.

3.3 Functional Flow Chart (Major Subroutine)

The Functional Flow Chart shown in Figure 4 is similar (but not identical) to the Program Flow Diagram of Figure 2, which is designed to illustrate the logical method by which the number 13 Height Processing Subroutine was implemented. Subroutines within the number 13 subroutine are identified with regard to their logical implementation. An annotated listing (not shown) is also required to further define the manner of logic implementation. This documentation provides a method for identifying and controlling program changes so that only the specific program steps requiring changes are affected.

FIGURE 1
EXAMPLE OF PROGRAM FLOW DIAGRAM;
MAJOR ROUTINE



PROGRAM FLOW DIAGRAM FOR HEIGHT PROCESSING SUBROUTINE

FIGURE 2

EXAMPLE OF PROGRAM FLOW DIAGRAM; MAJOR SUBROUTINE

13. Height-Processing Subroutine

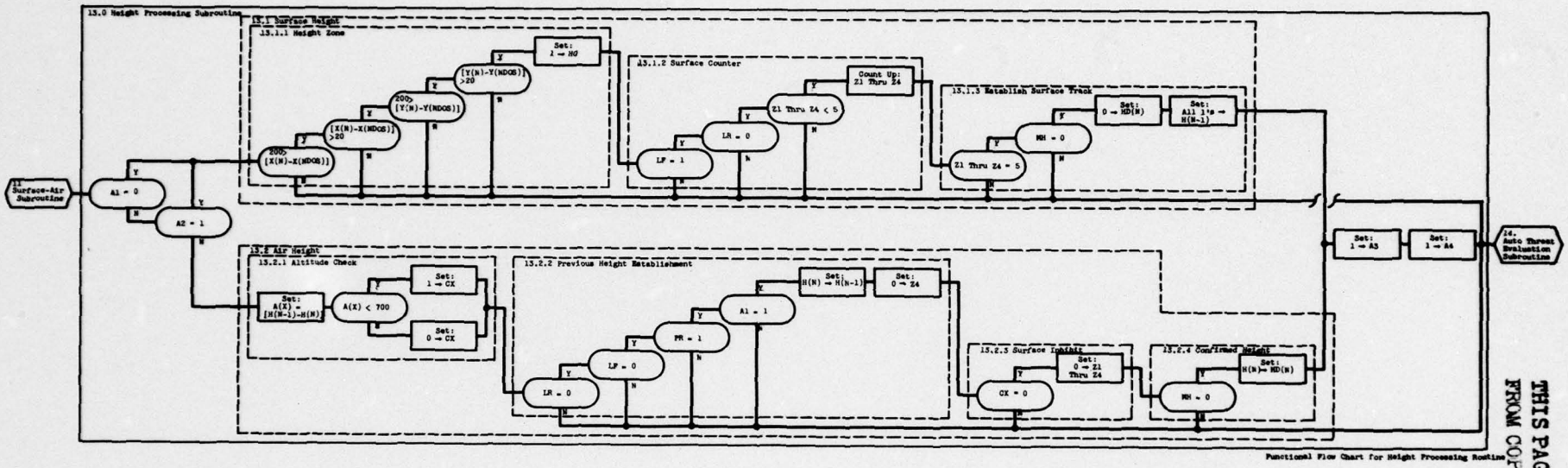
The computer-detector performs the height-finding process by measuring the time-difference between the direct path and the multi-path return echo. Results of the CD test are stored in the Height Accumulator. In addition to the multipath target feature, other criteria are established for Height Processing. Conditions are provided in Figure 2, 13.2.

Sub-Subroutines

13.1	Surface Height	Confirmed surface targets (0 height) are processed in this grouping (Figure 2, 13.1). Requirements for confirmed surface targets are as follows: (a) Target not locked with front tag (b) Target not locked with rear tag (c) No manual height entered (d) Valid one echo report (e) Surface counter indicates five valid single echo returns
13.1.1	Height Zone	In this section the target is tested for position within the height-finding zone. The height-finding zone is defined as further than 20 miles, but closer than 200 miles. When the criteria is met, the target in Height Finding Zone Bit is set as shown in Figure 2, 13.1.1.
13.1.2	Surface Counter	Rules for counting valid echo for surface counter are: (a) Ignore targets in obscuration zone (front or rear tag set). (b) Increment valid echos. In this grouping the front and rear tag are tested for target location outside obscuration zone, the surface counter is tested to verify that the surface count is less than 5, and then the surface counter is incremented by one as shown in Figure 2, 13.1.2. Subsequently, the surface counter is tested for being equal to five.
13.1.3	Establish Surface Track	In this section the surface counter is tested for adequate correlation of valid surface echos (surface counter equal to five) and, provided that manual height has not been entered, establishes the target as a surface target by setting present height to 0 and previous height to maximum. This ensures a difference between present height and previous height greater than 700 feet, thus preventing false returns from establishing this track as an air track as shown in Figure 2, 13.1.3.

FIGURE 3

FUNCTIONAL DESCRIPTION OF LOGIC FLOW FOR SUBROUTINE 13



Functional Flow Chart for Height Processing Routine

FIGURE 4

EXAMPLE OF FUNCTIONAL FLOW CHART;
MAJOR SUBROUTINE

4. CONCLUSIONS

Computer program documentation prepared in accordance with ARINC Research Publication 414-04-4-692 provides extensive improvements in:

- Program visibility
- Program identification
- Program understanding
- Identification of logic implementation and change control

5. RECOMMENDATIONS

It is recommended that:

- Computer program documentation for new tactical data systems be procured in accordance with the requirements of ARINC Research Publication 414-04-4-69
- Computer program documentation for existing tactical data systems be augmented with Program Flow Diagrams and Functional Flow Charts from the publication cited above, particularly in areas of:
  - Low reliability
  - High maintenance requirements
  - Frequent occurrence of changes
  - Exceptionally inadequate existing documentation

APPENDIX

CONVENTIONAL FLOW CHARTS FOR A HEIGHT-PROCESSING SUBROUTINE

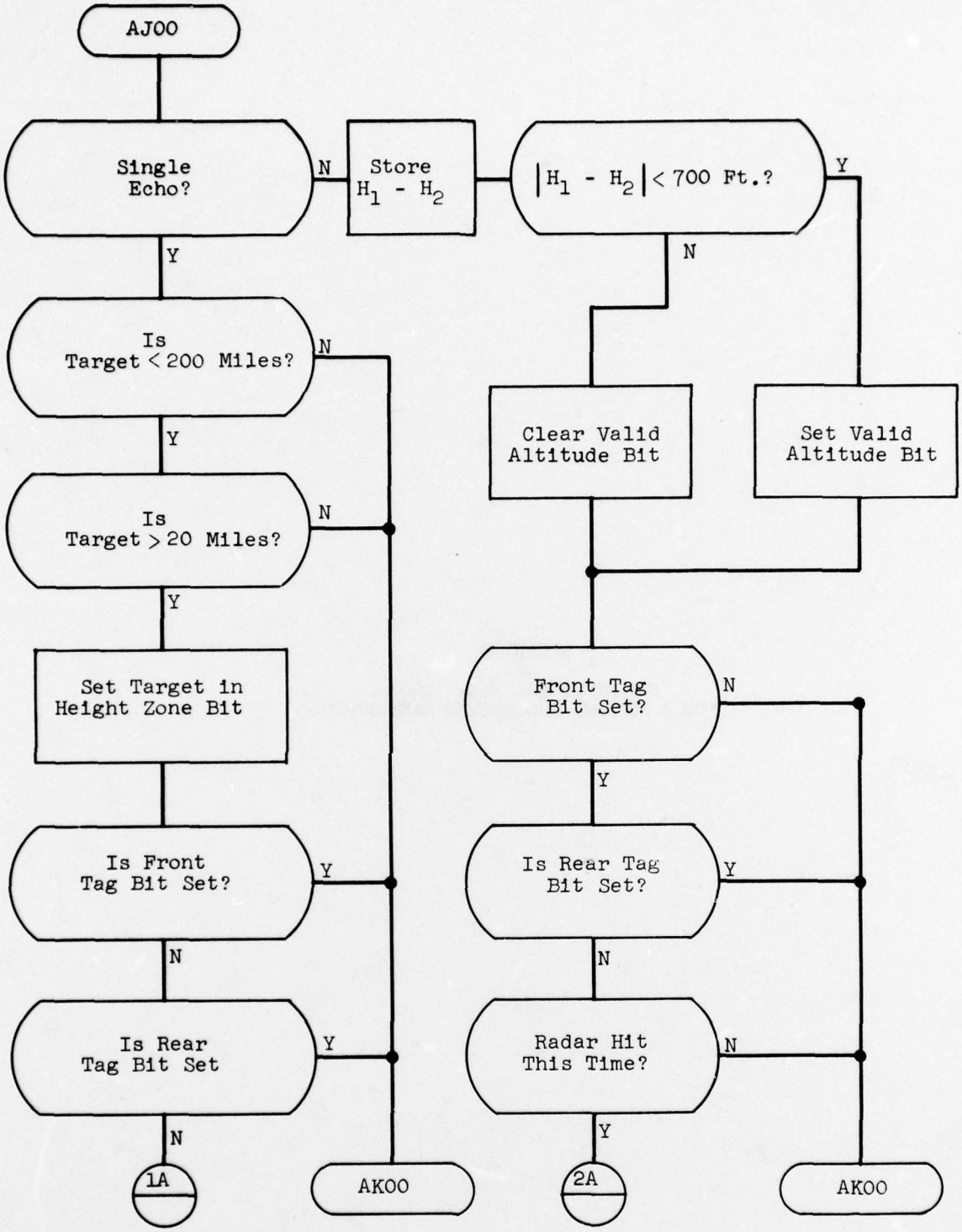


FIGURE A-1

FLOW CHART FOR HEIGHT-PROCESSING SUBROUTINE

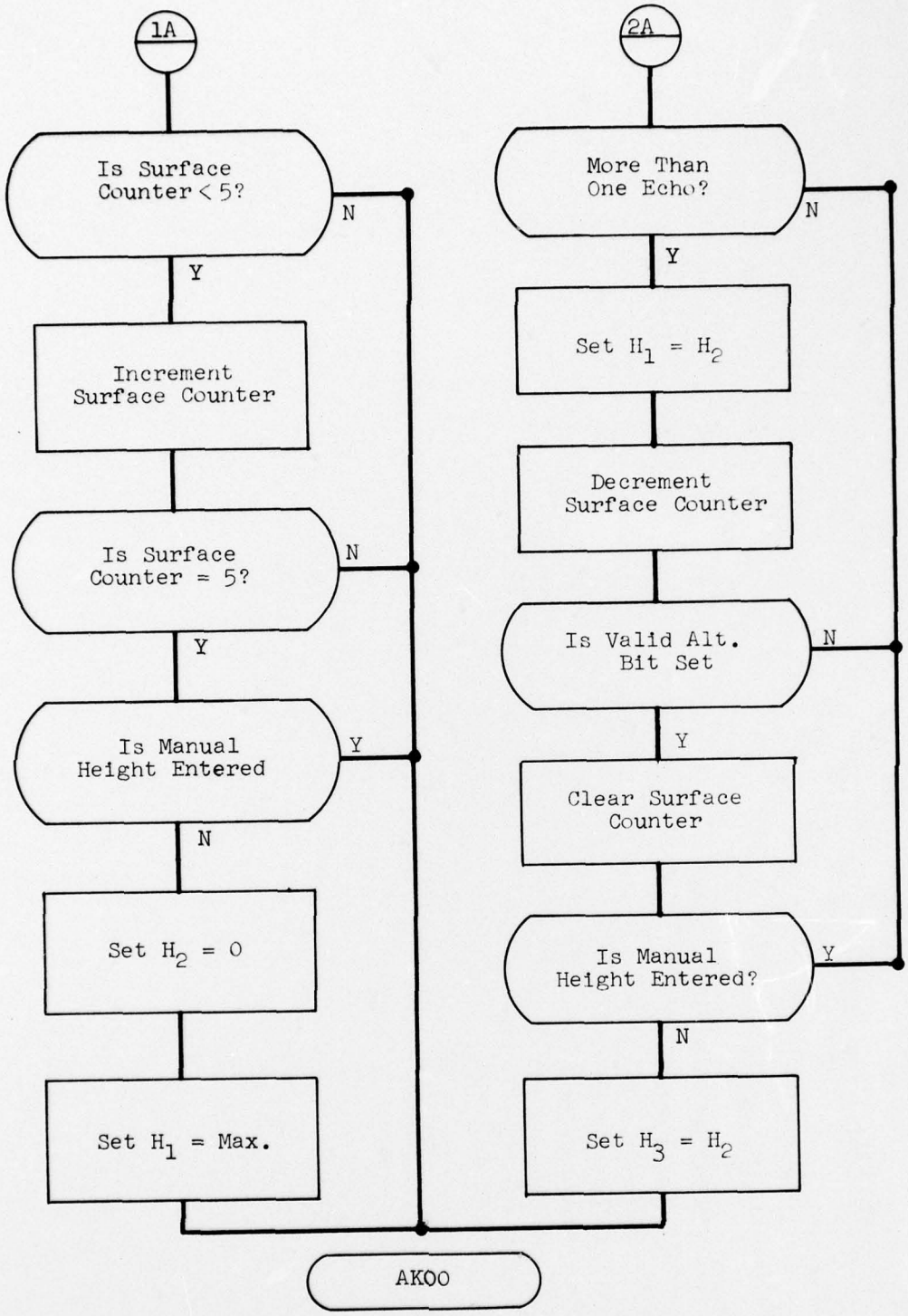


FIGURE A-1 (continued)